  
UNIVERSITY OF HERTFORDSHIRE  
School of Physics, Engineering and Computer Science

MSc Cyber Security  
[7COM1039 - Advanced Computer Science Masters Project](https://herts.instructure.com/courses/121503/modules/items/4405384)  
13 July 2025

Securing APIs for Cloud Services: Implementation and Security Evaluation Using Free Tools

Name: Ankit Joshi  
Student ID: 22032491  
Supervisor: Dr. Eric Chiejina

**1 Introduction and overview**

The digital transformation sweeping across industries has irrevocably positioned Application Programming Interfaces (APIs) as the foundational backbone of modern cloud-based services and interconnected applications. From mobile banking to e-commerce platforms and IoT devices, APIs facilitate seamless data exchange and functionality, enabling the agility and innovation characteristic of the contemporary digital economy (Nandedkar A, 2025). However, this ubiquitous integration has simultaneously rendered APIs a primary and increasingly attractive target for cyberattacks (Akamai, 2024). The escalating threat landscape is starkly underscored by recent industry reports, which reveal a significant and alarming trend: a substantial percentage of organizations are experiencing API-related data breaches, with billions of API attacks recorded globally within recent periods. This pervasive and growing threat unequivocally highlights an urgent and critical need for robust, proactive, and adaptable API security measures (Thales Group, 2024).

The nature of API vulnerabilities is multifaceted, often stemming from design flaws, misconfigurations, or inadequate implementation of security controls. The OWASP API Security Top 10, a widely recognized standard, consistently identifies critical risks such as Broken Object Level Authorization, Broken User Authentication, Excessive Data Exposure, and Security Misconfiguration, among others (OWASP Foundation, 2023). These vulnerabilities are not merely theoretical; they translate directly into tangible risks, including data theft, service disruption, and reputational damage, making API security a paramount concern for any entity operating in the cloud.

While the imperative for strong API security is universal, the capacity to implement sophisticated defenses is not evenly distributed across the organizational spectrum (Madupati B, 2023). Large enterprises typically possess the substantial financial resources and dedicated cybersecurity personnel required to invest in and manage complex commercial security solutions. These solutions often encompass advanced API gateways, dedicated Web Application Firewalls (WAFs) with API-specific policies, specialized API security platforms offering real-time threat detection, and comprehensive security information and event management (SIEM) systems. Such investments enable a multi-layered defense strategy, continuous monitoring, and rapid incident response capabilities, tailored to their extensive digital footprints and high-value assets (Sindall G, 2024).

In stark contrast, Small and Medium-sized Enterprises (SMEs) and independent developers face a distinct set of formidable challenges in securing their cloud-hosted APIs. These challenges are fundamentally rooted in significant resource constraints, including:

* **Limited Cybersecurity Budgets:** SMEs often operate with lean budgets, making the procurement of expensive commercial API security tools financially unfeasible. This forces them to seek more cost-effective alternatives, which may not always offer the same breadth or depth of protection (Verizon, 2025).
* **Constrained In-house Expertise:** Many SMEs lack dedicated cybersecurity teams or specialists with deep knowledge of API security best practices and emerging threats. Existing IT staff may have broad responsibilities, limiting their capacity to focus solely on specialized security domains.
* **Underestimated Vulnerability Perception:** Despite the reality that 43% of cyberattacks specifically target small businesses, only a small fraction of these entities are adequately prepared (NinjaOne, 2025). This often stems from a misconception that they are not attractive targets for sophisticated attackers, leading to insufficient investment in security infrastructure and training.
* **Complexity of Cloud Security:** Navigating the intricacies of cloud security configurations, even within free-tier services, can be daunting for organizations without specialized cloud security architects. Ensuring proper authentication, authorization, network segmentation, and data encryption across cloud services requires specific knowledge and continuous vigilance.

This disparity creates a critical industry demand for accessible, effective, and truly low-cost API security strategies. Such strategies must leverage readily available free tiers of cloud providers and robust open-source tools, providing a viable pathway for resource-constrained entities to achieve an acceptable level of security without prohibitive costs.

Academically, the field of API security has seen extensive research, broadly categorized into several areas. Much of the existing literature focuses on general API security principles and best practices, with the OWASP API Security Top 10 serving as a foundational reference (OWASP Foundation, n.d.). This body of work provides comprehensive guidance on common vulnerabilities and high-level mitigation strategies. Furthermore, significant research has been dedicated to enterprise-grade security solutions and frameworks within cloud computing environments ((Ali, Khan and Vasilakos, 2015; Atlidakis, Godefroid and Polishchuk, 2020). These studies often explore advanced security architectures, sophisticated threat detection mechanisms, and compliance with stringent regulatory standards relevant to large organizations. Research also exists on individual open-source tools and their capabilities in specific security testing scenarios (Hadi & Nugroho, 2020; Hadjimichael & Mitropoulos, 2023; Krasniqi, 2018). For instance, studies have explored the effectiveness of tools like OWASP ZAP in identifying common web application vulnerabilities, and the utility of manual penetration testing tools in uncovering complex logical flaws. There is also discourse around the inherent security risks associated with open-source software (SentinelOne, 2025), which is a relevant consideration when relying on such tools.

However, despite this breadth of research, a significant and critical knowledge gap persists: there is a notable dearth of comprehensive, empirically-driven research specifically evaluating the combined efficacy and practical limitations of *solely* free-tier cloud-native services and open-source tools in mitigating common API vulnerabilities within a cloud environment relevant to SMEs (Raidiam, 2025). While individual components (e.g., Azure free-tier features for authentication, TLS, basic rate limiting, and OWASP ZAP for vulnerability scanning) are well-documented in isolation, their synergistic effectiveness, ease of implementation when combined, and overall suitability as a consolidated, budget-friendly security framework for non-enterprise use cases remain largely underexplored. Existing studies rarely provide an integrated, empirical assessment of what can realistically be achieved within a 'zero-budget' or 'minimal-budget' security model for cloud-hosted APIs. This project directly addresses this critical gap. By developing a vulnerable API on Azure's free tier, implementing free-tier cloud-native controls, and then systematically testing its security posture using open-source tools, this research aims to provide concrete, empirical evidence. This evidence will not only contribute significantly to the academic understanding of 'security by resource constraint' but will also offer a valuable, evidence-based guide and practical recommendations for resource-constrained entities seeking to enhance their API security posture effectively and affordably.

**1.2 Problem statement**

The primary problem addressed by this project is the lack of empirically-driven research evaluating the combined efficacy and practical limitations of solely free-tier cloud-native services and open-source tools in mitigating common OWASP API Top 10 vulnerabilities within a cloud environment, specifically for resource-constrained Small and Medium-sized Enterprises (SMEs) and independent developers.

**1.3 Aim**

To evaluate the effectiveness of free-tier cloud-native and open-source security tools in mitigating OWASP API Top 10 vulnerabilities in cloud-hosted APIs, with a focus on accessible, low-cost security solutions suitable for small businesses and independent developers. This project seeks to provide evidence and practical recommendations for secure API development and deployment within resource-constrained environments.

**1.4 Objectives**

To achieve the project aim and answer the research questions, the following objectives will be pursued:

* **Develop a Cloud-Hosted API:** Design and implement a RESTful API deployed on Microsoft Azure's free-tier services, focusing on a simple use case that allows for the introduction of common vulnerabilities.
* **Implement Essential Security Controls:** Integrate and configure key security controls within the API and its Azure hosting environment, specifically focusing on free-tier cloud-native features for Authentication & Authorization, Transport Layer Security (TLS), Rate Limiting, and Input Validation & Error Handling.
* **Conduct Comprehensive Vulnerability Assessments:** Perform security testing against the deployed API using established open-source tools and methodologies, primarily OWASP ZAP, complemented by manual penetration testing techniques targeting OWASP API Top 10 vulnerabilities.
* **Compare Effectiveness:** Systematically compare the effectiveness of the implemented free-tier cloud-native versus open-source tools based on quantifiable criteria such as vulnerability detection rate, severity reduction, ease of deployment, scalability considerations, and identified limitations.
* **Contextualize Findings & Derive Recommendations:** Analyse the results by mapping them to existing API security frameworks and literature (e.g., OWASP, NIST SP 800-53) and derive practical, evidence-based security recommendations for SMEs and independent developers.

**1.5 Tools and Techniques**

* **Cloud Platform**: Microsoft Azure Free Tier
* **Framework**: FastAPI (Python)
* **Security Controls**: Azure Entra ID (OAuth2), TLS, rate limiting via slowapi, Pydantic for input validation
* **Testing Tools**: OWASP ZAP, Postman/Insomnia, Burp Suite Community Edition

**1.6 Deliverables**

* Design and source code for the vulnerable API
* Security control configurations (JSON, YAML, Python code)
* Test plans and ZAP scan reports
* Analysis of vulnerabilities detected and their severity
* Final report and presentation materials

**1.7 Consideration of Ethical, Legal, Professional, and Social Issues**

This project inherently deals with security vulnerabilities and testing, necessitating a rigorous consideration of ethical, legal, professional, and social implications.

* **Ethical Issues:**
  + **No Interaction with Live Systems:** All testing is conducted strictly within an isolated, simulated environment on Azure's free tier. This ensures that no live production systems or third-party APIs are interacted with or potentially harmed, upholding the ethical principle of "do no harm."
  + **Data Privacy:** Crucially, this project operates exclusively with simulated, non-identifiable data for testing purposes, ensuring that no real user data is ever processed or stored. Therefore, privacy concerns related to personal data are entirely mitigated within the scope of this project.
  + **Responsible Disclosure:** Should any critical vulnerabilities be discovered in the tools themselves (though unlikely for established open-source tools), a responsible disclosure process would be followed.
* **Legal Issues:**
  + **Data Protection Regulations (GDPR/DPA):** Although this project utilizes **simulated, non-identifiable data**, it is designed with a strong awareness of compliance with data protection regulations such as General Data Protection Regulation (European Union, 2016) and the Data Protection Act 2018 (UK Government, 2018) for any real-world deployment. The implemented security controls (authentication, authorization, input validation) contribute to a secure data processing environment, which is a key aspect of compliance.
  + **Intellectual Property:** The project utilizes open-source tools (OWASP ZAP, Postman/Insomnia, Burp Suite Community Edition) and free-tier cloud services. Adherence to their respective licensing agreements (e.g., Apache 2.0, MIT License) is maintained
* **Professional Issues:**
  + **Adherence to Best Practices:** The project adheres to industry best practices for API security, primarily the OWASP API Security Top 10, and aims to provide recommendations aligned with frameworks like NIST SP 800-53.
  + **Quality Assurance and Documentation:** Emphasis is placed on developing robust, well-documented code and configurations, as well as maintaining comprehensive test records.
  + **Responsible Innovation:** The project aims to provide practical, evidence-based recommendations, promoting responsible and secure API development, particularly for resource-constrained entities.
* **Social Issues:**
  + **Empowering SMEs:** The core social contribution of this project is to democratize API security. By providing accessible, low-cost solutions, it aims to empower SMEs and independent developers to better protect their cloud services, thereby reducing their vulnerability to cyberattacks and fostering a more secure digital ecosystem.
  + **Digital Equity:** Contributing to accessible security knowledge can help bridge the digital divide for smaller organizations that cannot afford expensive commercial solutions.

**Knowledge of University Ethics Approval:** The project, "Securing APIs for Cloud Services: Implementation and Security Evaluation Using Free Tools," does not involve human subjects, personal data collection, or interaction with live production systems. All testing is confined to a controlled, simulated environment developed by the student. Therefore, direct ethical clearance from the University of Hertfordshire (UH) for human interaction is not required for the scope of this research. However, the project fully acknowledges and adheres to the university's overarching ethical guidelines for research, ensuring responsible conduct, data integrity (of simulated data), and the avoidance of harm. The project's methodology is designed to align with these principles, ensuring that all activities are conducted in a responsible and ethical manner.

**2. Progress to Date**

Significant progress has been made in the initial phases of the project, focusing on establishing the foundational cloud environment and developing a vulnerable RESTful API, followed by the implementation of essential security controls. This aligns with Phase 1 (Project Setup & Core API Development) and Phase 2 (Core Security Control Implementation) of the proposed methodology.

**Phase 1: Project Setup & Core API Development (Weeks 1-2)** During the initial two weeks, the project environment was successfully established on Microsoft Azure's free tier. This involved:

* **Azure Account Setup:** A new Azure subscription was configured, and essential free-tier resources were provisioned. This included setting up an Azure App Service instance to host the API and configuring a basic Microsoft Entra ID (formerly Azure AD) Free tenant for identity management.
* **RESTful API Implementation:** A core RESTful API was designed and implemented using **FastAPI** in Python. This API serves as a controlled environment for security evaluation, intentionally incorporating points for common vulnerabilities. It currently exposes two main functional endpoints:
  + **/secure-data (GET):** Designed to represent a sensitive data retrieval operation.
  + **/submit-data (POST):** Designed for data submission, requiring robust input handling. A root endpoint (/) is also available for basic API health checks.
* **Deployment to Azure App Service:** The developed FastAPI application was successfully deployed to the Azure App Service, ensuring it was accessible via a public endpoint.

*(See Appendix A: Figure A.1 for a screenshot of the Azure App Registration configuration and Figure A.2 for the API's initial endpoint structure)* *(See Appendix B: Code Snippet B.1 for the core Python FastAPI application code, specifically the endpoint definitions.)*

**Phase 2: Core Security Control Implementation (Weeks 3-4)** Following the API development, efforts shifted to integrating and configuring key security controls directly within the FastAPI application and its Azure hosting environment, leveraging free-tier capabilities:

* **Authentication and Authorization:** Azure Entra ID (formerly Azure AD) has been integrated using OAuth2AuthorizationCodeBearer for user authentication. The verify\_token function, leveraging jose library, validates JWT tokens issued by Azure AD, ensuring their authenticity and integrity by checking signatures against fetched JWKS keys. For **authorization**, the /secure-data endpoint explicitly implements role-based access control, allowing access only to users possessing the "Admin" role within their token payload.
* **Transport Layer Security (TLS):** HTTPS enforcement was configured at the Azure App Service level, ensuring all API communication is encrypted in transit. This was a straightforward configuration step within the Azure portal, leveraging the platform's built-in capabilities to provide secure communication channels.
* **Input Validation and Error Handling:** Robust server-side **input validation** has been implemented for the /submit-data endpoint using Pydantic models (SecureDataRequest). This ensures that incoming data (e.g., username, email, comment) conforms to predefined patterns, lengths, and types, preventing common injection attacks and malformed data submissions. Generic error responses are also handled by FastAPI's default mechanisms and custom exception handlers where appropriate, avoiding the exposure of sensitive internal system details during errors.
* **Rate Limiting:** **Rate limiting** has been implemented using the slowapi library to protect both the /secure-data and /submit-data endpoints from abuse. The /secure-data endpoint is limited to 5 requests per minute, while the /submit-data endpoint allows 10 requests per minute from the same IP address. A custom exception handler catches RateLimitExceeded errors, returning a 429 Too Many Requests response.

At this interim stage, the foundational FastAPI API is operational with robust initial security controls in place, directly reflecting the specified objectives. The project is now poised to move into the comprehensive vulnerability assessment phase. Preliminary manual tests confirm the basic functionality of the API and the initial enforcement of authentication, authorization, rate limiting, and input validation.

*(See Appendix A: Figure A.3 for a screenshot of Azure Entra ID integration)* *(See Appendix B: Code Snippet B.2 for examples of input validation logic and generic error handling,)*

**3. Planned Work**

The project is progressing according to the outlined methodology. Following the successful completion of initial setup and core security control implementation (Phases 1 and 2), the remaining work will focus on comprehensive vulnerability assessment, detailed analysis, and final reporting. The updated project plan, including remaining tasks and timelines, is detailed below.

**Remaining Phases:**

* **Phase 3: Targeted Vulnerability Assessment (Weeks 5-6)**
  + **Week 5:**
    - Install and configure OWASP ZAP (if not fully completed).
    - Familiarize thoroughly with OWASP ZAP's core API scanning features, including active and passive scanning, and API definition import (e.g., OpenAPI/Swagger).
    - Develop initial automated scan profiles within ZAP targeting common vulnerabilities.
  + **Week 6:**
    - Prioritize and develop concise manual test cases based on the OWASP API Security Top 10 (2023) relevant to the implemented API endpoints (e.g., testing for Broken Object Level Authorization, Broken User Authentication, Excessive Data Exposure).
    - Execute targeted automated ZAP scans against the API.
    - Perform critical manual penetration tests using Postman/Insomnia and Burp Suite Community Edition to identify complex logical flaws that automated tools might miss.
    - Document all key findings, including vulnerability types, severities (mapping to CVSS where possible), and specific locations/parameters affected.
* **Phase 4: Analysis, Improvements, and Reporting (Weeks 7-8)**
  + **Week 7:**
    - Collect and consolidate all empirical data from automated scan reports (OWASP ZAP) and manual penetration testing documentation.
    - Perform quantitative analysis: Compare vulnerability counts and severity reduction across different control phases (e.g., baseline API vs. API with Azure controls vs. API with open-source tool findings). Assess coverage against the OWASP API Top 10.
    - Perform qualitative analysis: Evaluate aspects such as ease of deployment, scalability, flexibility, and identified limitations or gaps in the free-tier tools.
    - Identify areas for potential security improvements based on analysis.
  + **Week 8:**
    - Rigorously contextualize findings by mapping them against established API security frameworks (OWASP API Security Top 10, NIST SP 800-53).
    - Derive practical, evidence-based security recommendations for SMEs and independent developers.
    - Draft the final project report, including introduction, literature review refinement, detailed methodology, results, discussion, conclusion, and future work.
    - Prepare appendices (final code, detailed test results, diagrams).

**Timeline:** The project is currently in Week 4, transitioning into Phase 3. The remaining weeks (5-8) are allocated for the vulnerability assessment, analysis, and final report writing, ensuring completion by the final submission deadline.

*(See Appendix D: Gantt Chart for detailed task breakdown)*

**Risk Management and Contingencies:** As identified in the DPP, potential risks include exceeding free-tier limits, unexpected tool behaviors, or configuration complexities. These are being mitigated through vigilant resource monitoring (using Azure Monitor), reliance on well-documented tools, and an iterative development and testing approach. In the event of significant unforeseen challenges, the scope of API complexity or the number of vulnerabilities tested can be adjusted to ensure the core research objectives are still met within the timeframe. For instance, if a specific tool proves unfeasible, an alternative open-source tool will be explored, or the focus will shift to manual testing for that specific vulnerability type.

**Bibliography**

**References**

Nandedkar, A. (2025) *The API economy: How APIs are shaping the future of digital transformation*. [online] Medium. Available at: https://aditya-nandedkar.medium.com/the-api-economy-how-apis-are-shaping-the-future-of-digital-transformation-eb601e99f85e [Accessed 13 Jun. 2025].

Akamai (2024) New study finds 84% of security professionals experienced an API security incident in the past year. [online] Akamai. Available at: https://www.akamai.com/newsroom/press-release/new-study-finds-84-of-security-professionals-experienced-an-api-security-incident-in-the-past-year [Accessed 13 Jun. 2025].

Thales Group (2024) Vulnerable APIs and Bot Attacks Costing Businesses Up to $186 Billion Annually. [online] Thales Group. Available at: https://cpl.thalesgroup.com/about-us/newsroom/vulnerable-apis-bot-attacks-costing-businesses-186-billion [Accessed 13 Jul. 2025].

OWASP (2023) OWASP API Security Project. Available at: https://owasp.org/www-project-api-security/ [Accessed 14 Jun. 2025].

Madupati, B. (2023) Comprehensive Approaches to API Security and Management in Large-Scale Microservices Environments. [online] SSRN. Available at: https://ssrn.com/abstract=5076630 [Accessed 15 Jun. 2025].

Sindall, G. (2024) 8 Enterprise API Security Strategy Best Practices. [online] DigitalML. Available at: https://www.digitalml.com/enterprise-api-security-strategy-best-practices/ [Accessed 17 Jun. 2025].

Innovate Finance (2025) New research reveals API security crisis: 84% of enterprises risk exposing sensitive data. [online] Innovate Finance. Available at: https://www.innovatefinance.com/blogs/new-research-reveals-api-security-crisis-84-of-enterprises-risk-exposing-sensitive-data/ [Accessed 6 Jul. 2025].

Das, I. (2024) Empowering Small Businesses in the Digital Age: A Must-Read Guide to Web Application & API Security. [online] Qualys. Available at: https://blog.qualys.com/product-tech/2024/04/22/empowering-small-businesses-in-the-digital-age-a-must-read-guide-to-web-application-api-security [Accessed 19 Jun. 2025].

Rahmonbek, K. (2025) 35 Alarming Small Business Cybersecurity Statistics for 2025. [online] StrongDM. Available at: https://www.strongdm.com/blog/small-business-cyber-security-statistics [Accessed 13 Jul. 2025].

Verizon (2025) 2025 Data Breach Investigations Report: Small- and Medium-Sized Business Snapshot. [online] Verizon. Available at: https://www.verizon.com/business/resources/infographics/2025-dbir-smb-snapshot.pdf [Accessed 20 Jun. 2025].

NinjaOne. (2025) 7 SMB Cybersecurity Statistics You Need to Know in 2025. Available at: https://www.ninjaone.com/blog/smb-cybersecurity-statistics/ (Accessed: 2 July 2025).

Ali, M., Khan, S.U. and Vasilakos, A. v. (2015) “Security in cloud computing: Opportunities and challenges,” Information Sciences, 305. Available at: https://doi.org/10.1016/j.ins.2015.01.025. [Accessed on 1 June 2025]

Atlidakis, V., Godefroid, P. and Polishchuk, M. (2020) “Checking Security Properties of Cloud Service REST APIs,” in Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020. Available at: https://doi.org/10.1109/ICST46399.2020.00046. [Accessed on 1 June 2025]

Asemi, Hadi (2023) A Study on API Security Pentesting Available at: https://www.proquest.com/openview/431b544974bbeb4cf446f1ec33fe341e/1?pq-origsite=gscholar&cbl=18750&diss=y [Accessed on 2 June 2025]

OpenAI, 2024. *ChatGPT* (version GPT-4). [online] Available at: https://chat.openai.com [Accessed: 13 July 2025].

Traceable AI. (2025). 2025 Global State of API Security. Available at: https://www.traceable.ai/wp-content/uploads/2024/10/2025-Global-State-of-API-Security.pdf [Accessed: 1 July 2025].

Google, 2024. *Gemini* (Generative AI Model). [online] Available at: https://gemini.google.com [Accessed 13 Jul. 2025].

European Union, 2016. *General Data Protection Regulation (GDPR) Regulation (EU) 2016/679 of the European Parliament and of the Council*. [online] Available at: https://eur-lex.europa.eu/eli/reg/2016/679/oj [Accessed 16 Jun. 2025].

UK Government, 2018. *Data Protection Act 2018*. [online] Available at: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted [Accessed 16 Jun. 2025].

Akamai. (2025). API Security Impact Study 2025. MySecurity Marketplace. Available at: https://mysecuritymarketplace.com/reports/api-security-impact-study-2025/ [Accessed: 1 July 2025].

Raidiam. (2025). Raidiam research: 84% of enterprises risk sensitive data exposure due to API security gaps. Intelligent CISO. Available at: https://www.intelligentciso.com/2025/07/02/raidiam-research-84-of-enterprises-risk-sensitive-data-exposure-due-to-api-security-gaps/ [Accessed: 1 July 2025].

Gupta, D., Singh, N. and Sharma, M. (2024) Cloud Security Challenges and Solutions: A Review of Current Best Practices. ResearchGate. Available at: https://www.researchgate.net/publication/387558426\_Cloud\_Security\_Challenges\_and\_Solutions\_A\_Review\_of\_Current\_Best\_Practices [Accessed: 1 July 2025].

Hadi, A. J. and Nugroho, S. (2020) 'Security analysis of REST API using OWASP ZAP and Postman', Journal of Physics: Conference Series, 2094(3), p. 032016. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/2094/3/032016/pdf [Accessed: 1 July 2025].

Hadjimichael, L. and Mitropoulos, K. (2023) API Security Testing: The Challenges of Security Testing for Restful APIs. ResearchGate. Available at: https://www.researchgate.net/publication/371174422\_API\_Security\_Testing\_The\_Challenges\_of\_Security\_Testing\_for\_Restful\_APIs [Accessed: 1 July 2025].

Krasniqi, F. (2018) Improving Web Application Security Using Open-Source Tools. Bachelor's thesis. University of Gävle. Available at: https://core.ac.uk/reader/542974100 [Accessed: 2 July 2025].

SentinelOne. (n.d.) Open Source Software Security Risks. Available at: https://www.sentinelone.com/cybersecurity-101/cybersecurity/open-source-software-security-risks/ [Accessed: 1 July 2025].

**Appendices**

**Appendix A: Screenshots of Azure Environment and API**

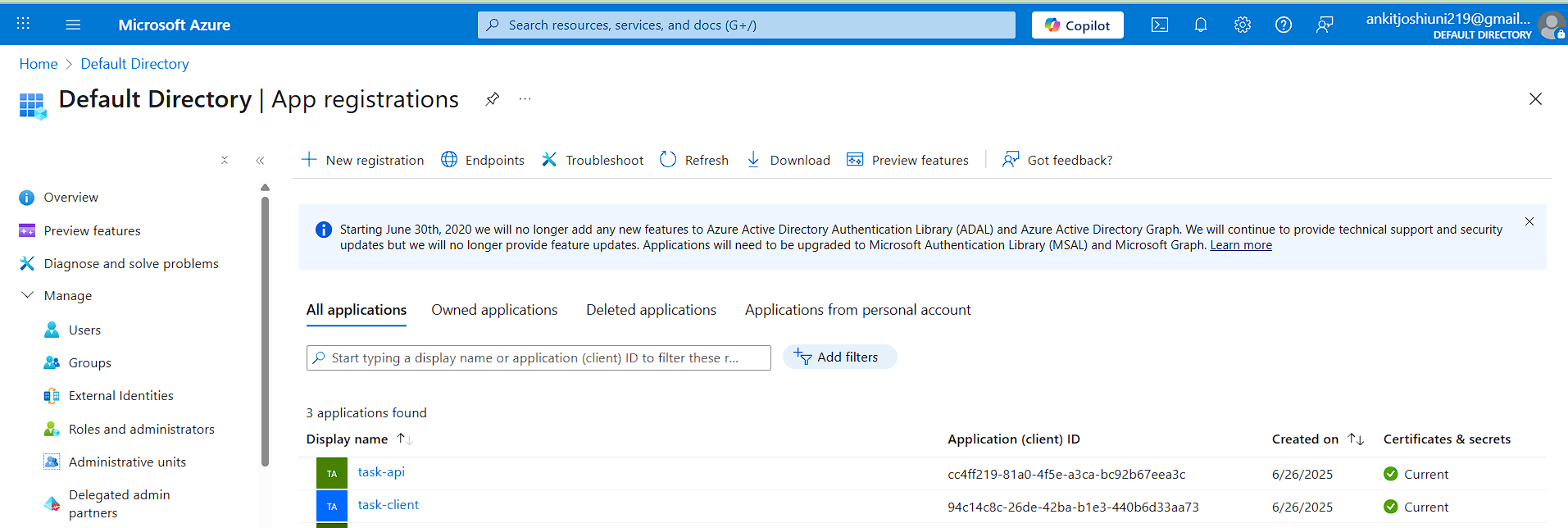


Figure A.1: Azure App Registration Configuration

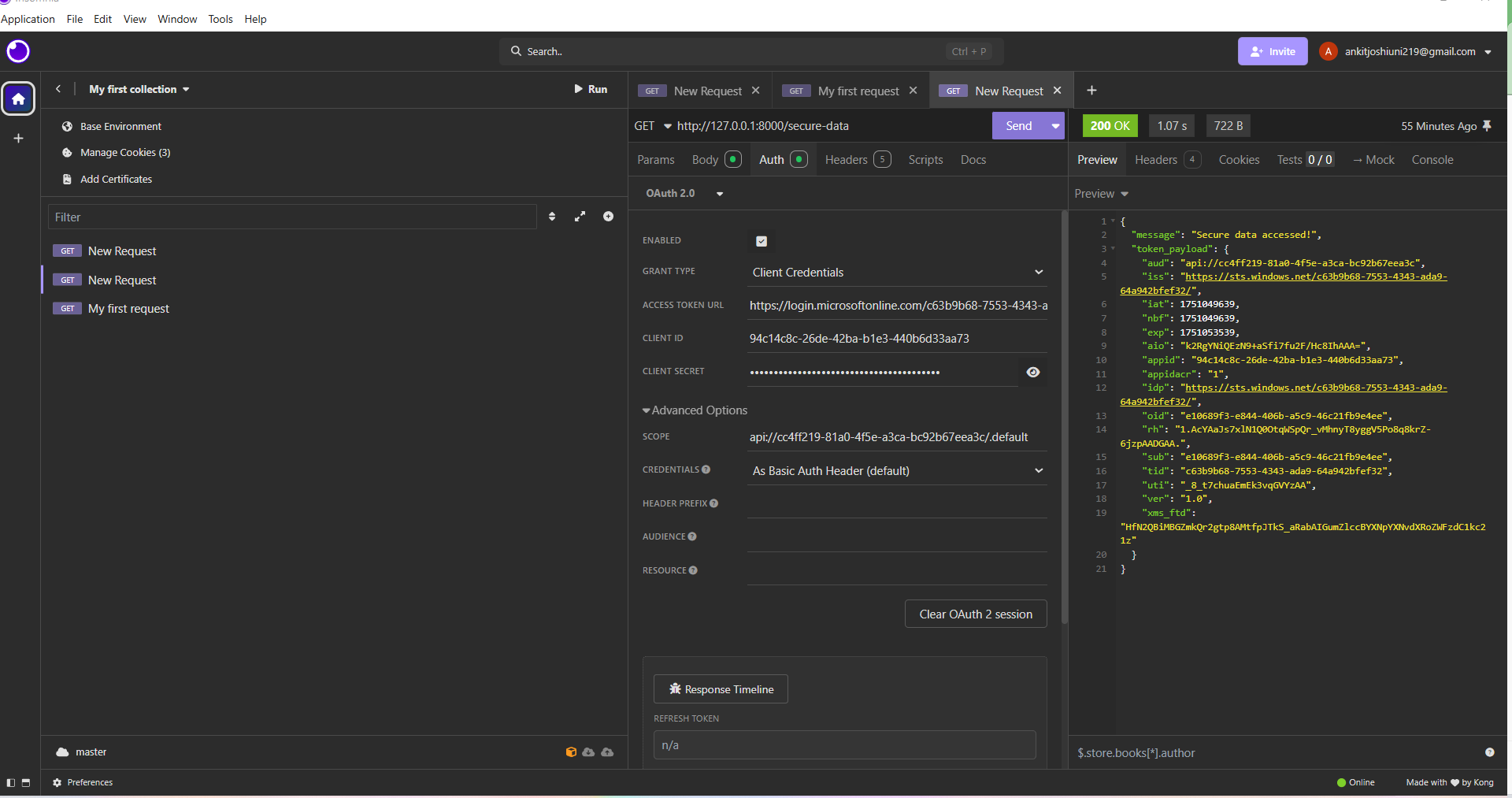
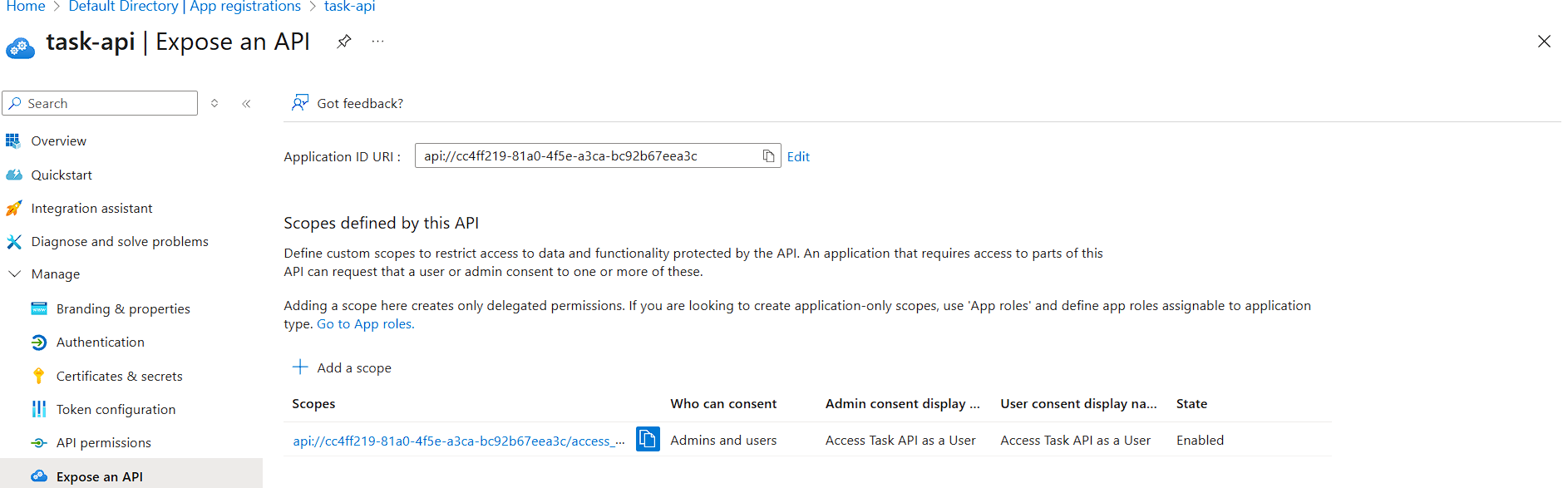


Figure A.2: API's Initial Endpoint Structure (Insomnia collection screenshot)



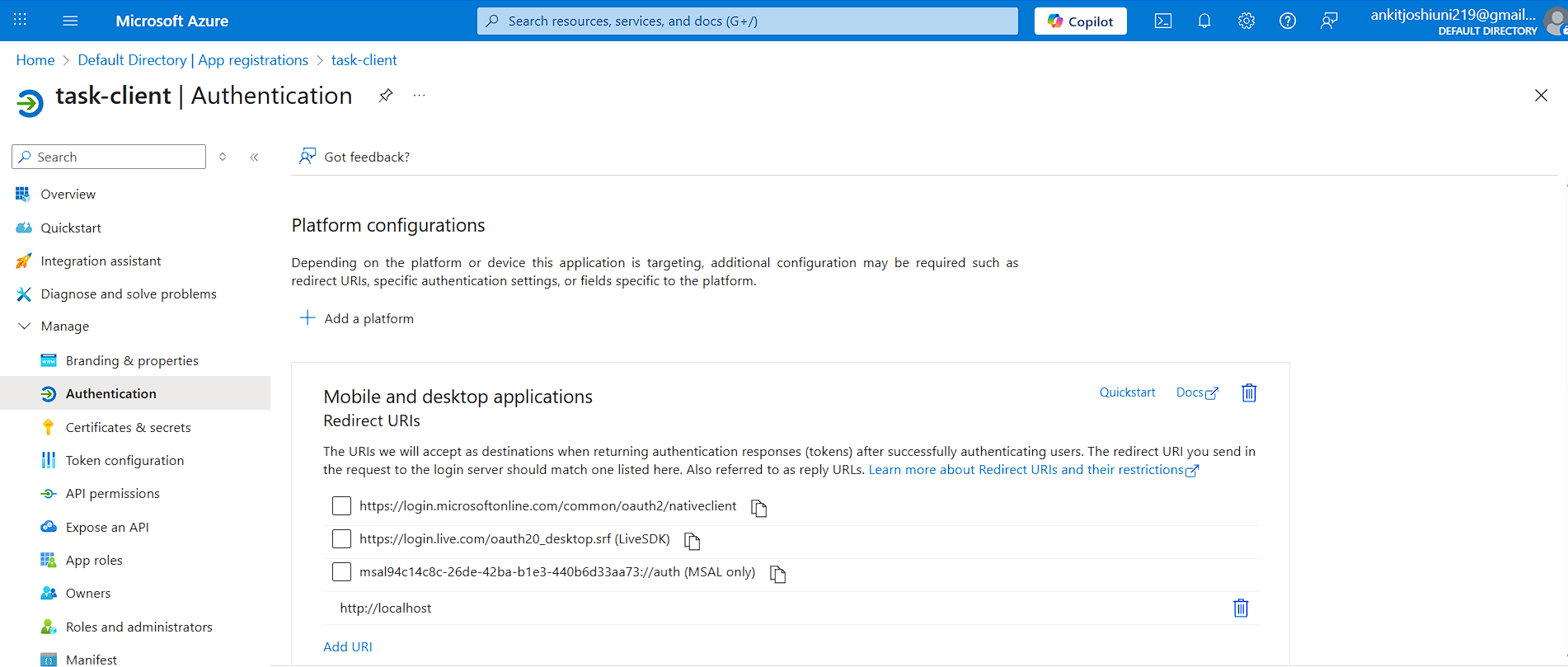
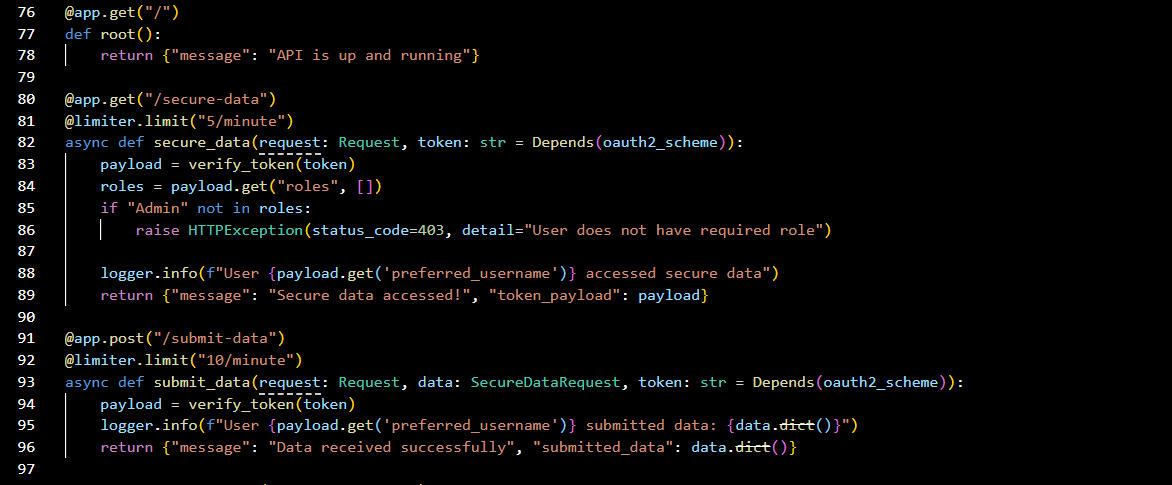
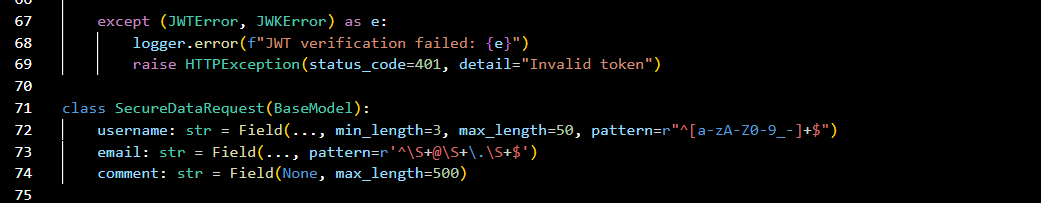


Figure A.3: Azure Entra ID Integration Screenshot

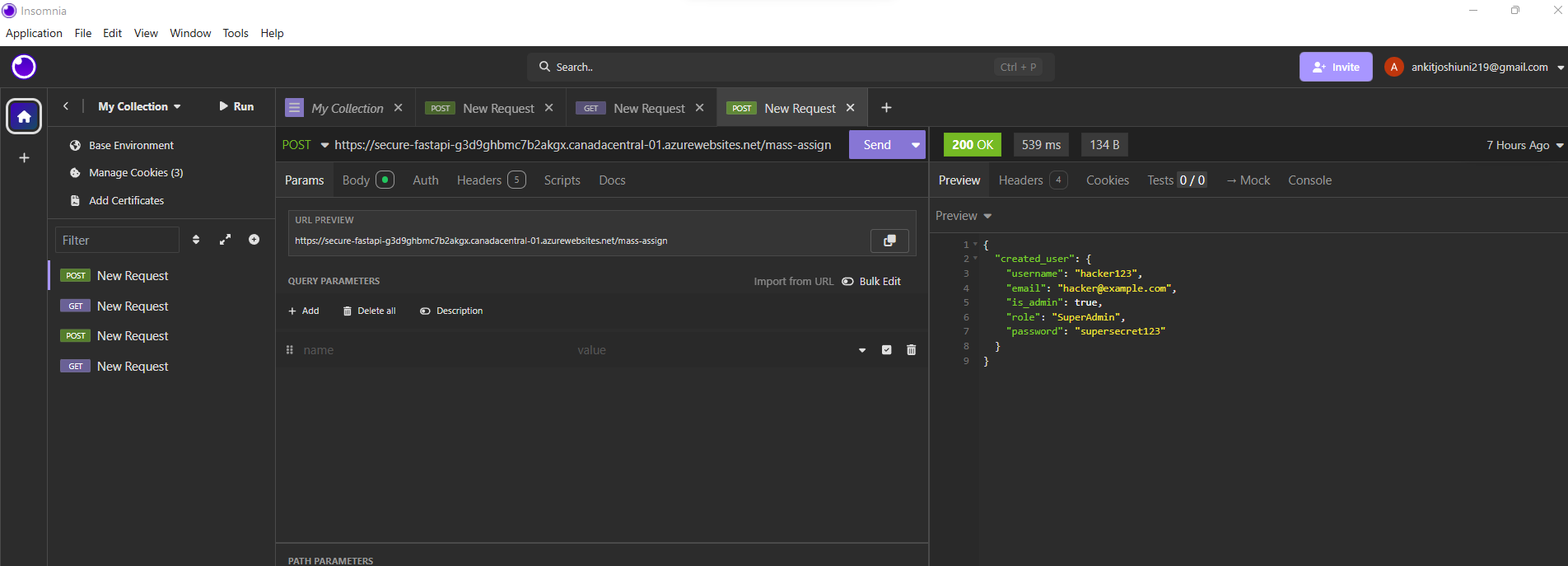
**Appendix B: Key Source Code Snippets**



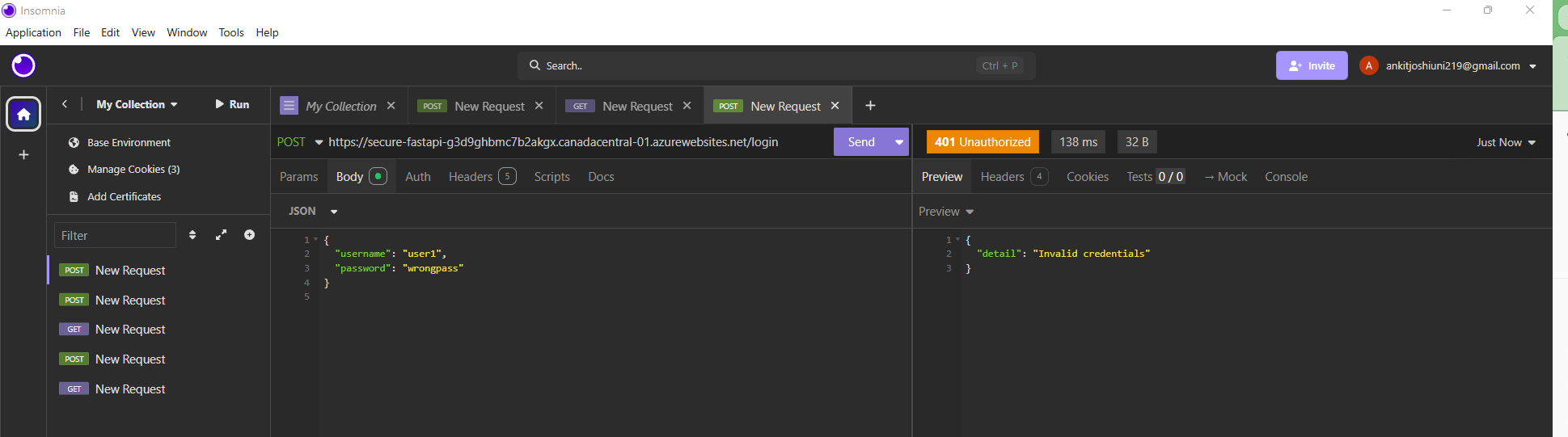
Code Snippet B.1: Core Python API Endpoint Definitions

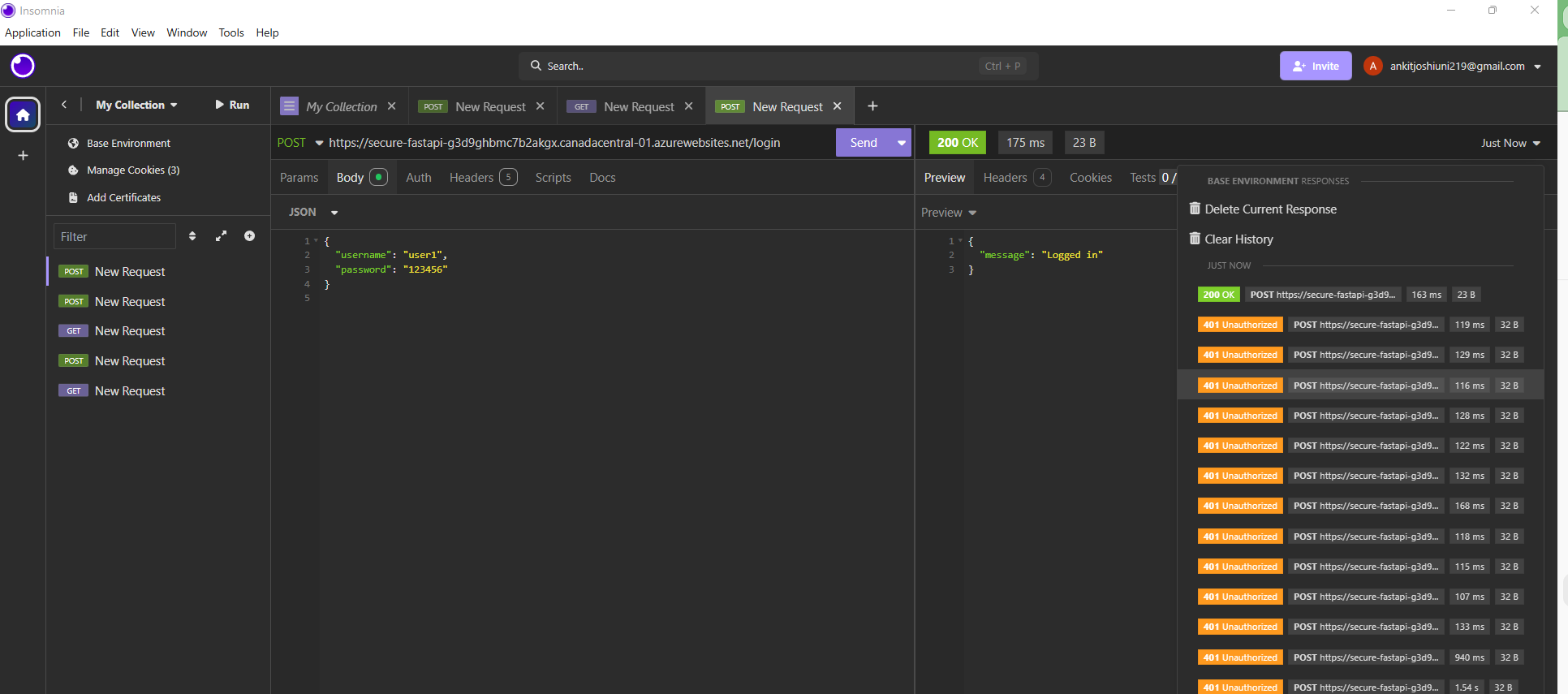


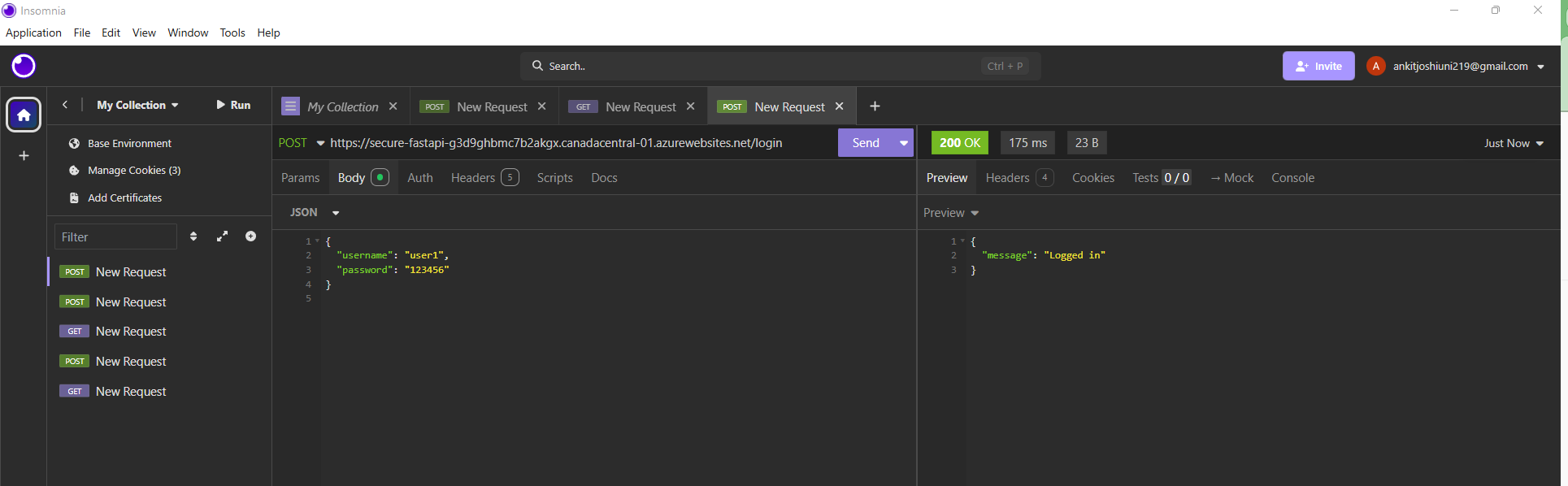
Code Snippet B.2: Examples of Input Validation Logic and Generic Error Handling

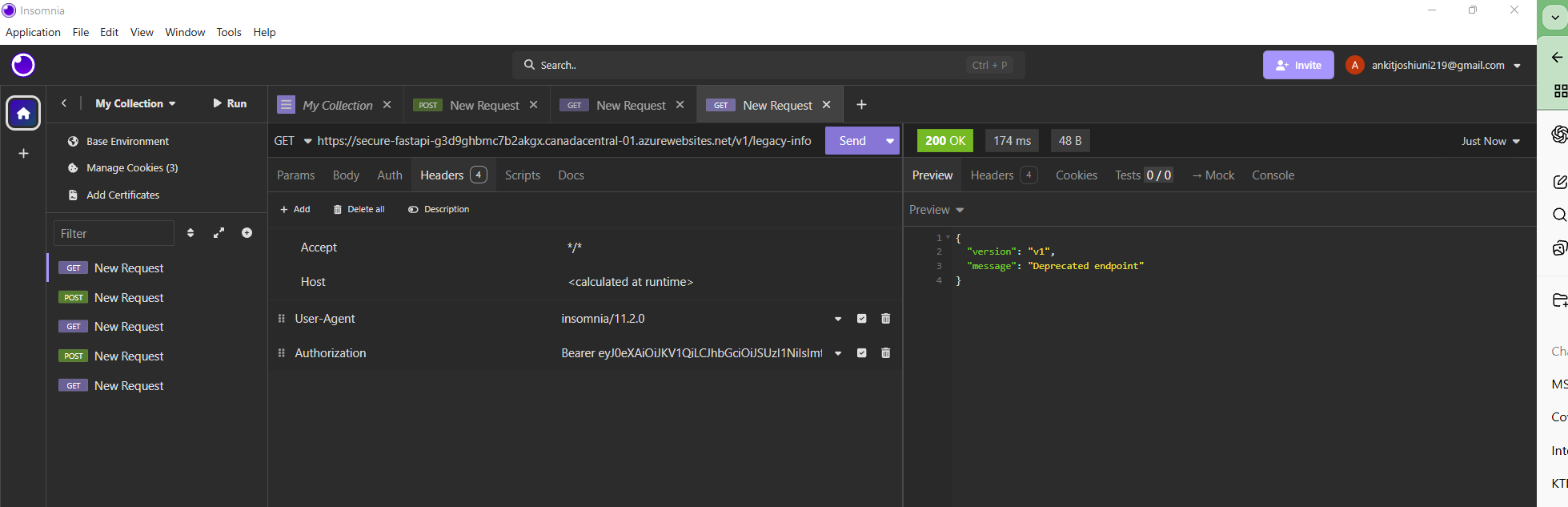


Api#6

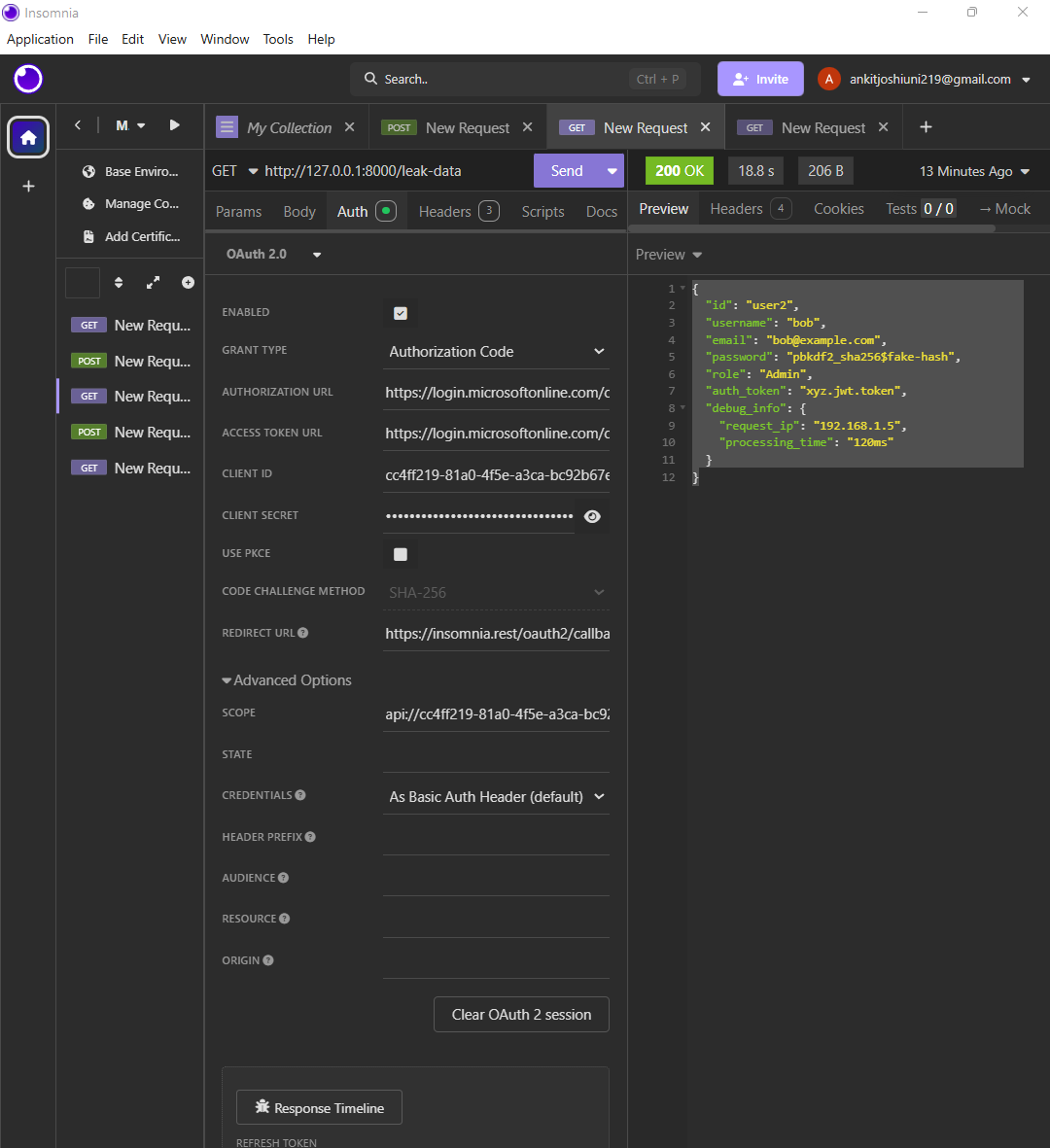
****

**Login Broken Auth Test**

****

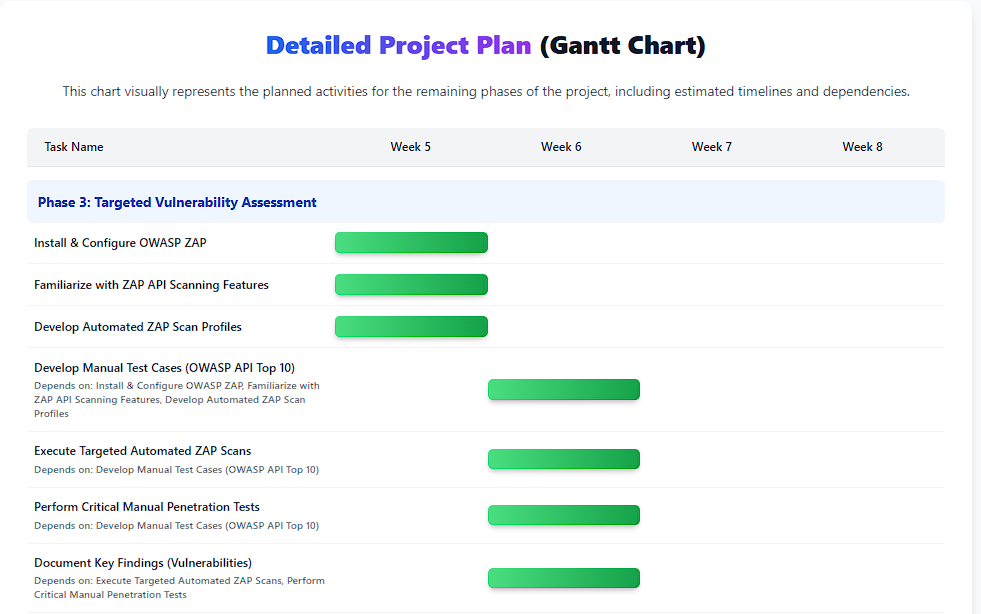
****

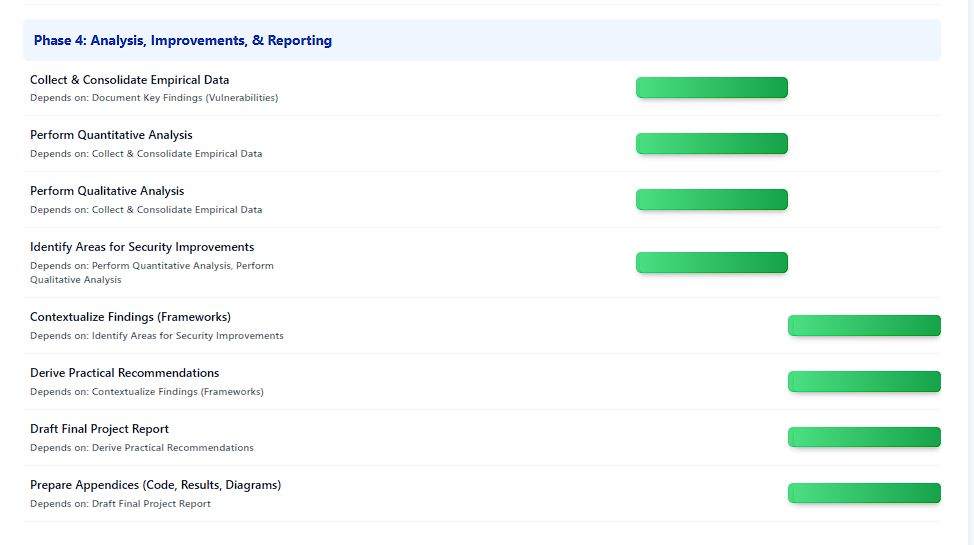
**deprecated legacy API**

****

**Excessive Data Exposure vuln from /leak-data.**

**Appendix D: Detailed Project Plan (Gantt Chart)**





Gantt Chart illustrating remaining tasks and timelines. (Google Gemini,2025)